International Conference on Applied Internet and Information Technologies, 2018
DOI:10.20544/AIIT2018.P15

Designing Intelligent Tutoring System Based on Bayesian Network

Mihajlo Hasanu*, Natasha Blazeska-Tabakovska**
*Faculty of Information and Communication Technologies, Bitola, Republic of Macedonia
**Faculty of Information and Communication Technologies, Bitola, Republic of Macedonia
mihajlo_hasanu@yahoo.com, natasa.tabakovska@fikt.edu.mk

Abstract— Nowadays, the educational system in each country is facing new challenges and recent IT innovations, which are the key factors for the success and quality of education that will result in a more efficient and effective knowledge transfer. Intelligent tutoring systems - ITS are a perfect IT solution that can contribute to overcome the restrictions imposed by the traditional teaching and improve the educational process. Guided by the need to improve education, the main purpose of this paper is to offer a model of architecture of an ITS for learning C# programing language – CPLITS. The system is intelligent because it has implemented an artificial technique called Bayesian Networks in its architecture. The aim of CPLITS is to adapt to the different needs of different categories of students through a mechanism that enables them support in the navigation through the online learning materials, support for delivering effective and efficient pedagogical instructions in the learning process while solving a particular problem as well as structuring of the teaching curriculum based on the student's profile.

Keywords— intelligent tutoring systems, model of intelligent tutoring systems, Bayesian network

I. Introduction

Studies of student learning have long shown that learning can be more effective if it is performed through private tutoring rather than teaching in a classroom [1]. In classroom teaching, all the students have to listen to the same lectures regardless of the knowledge that they have and theirs learning preferences. It is not possible for the teacher to deliver a lecture that is customized to each student's existing knowledge. An Intelligent Tutoring System (ITS) can address this problem. The ITS will enable the student to learn the material at their own learning speed, and can teach the materials that are relevant to the students' current knowledge and provide help that is specific to the students' problem [2]. By its nature, ITS can be defined as computer software systems that try to mimic the methods and dialogs of natural human tutors, to generate teaching activities in real time or at the request of different categories of students [3]. Over the years, ITS have been a subject of continuous changes and improvements evolving from computer aided instruction into intelligent computer aided instruction, today known as ITS, using various AI techniques such as: Bayesian networks, intelligent agents, fuzzy logic, neural networks, ontologies and many more that makes this system intelligent and more sophisticated.

The main goal of this paper is to present a design approach of an intelligent tutoring system that will adapt to the different needs of different categories of students through a mechanism that enables them support in the navigation through the online learning materials, support for delivering effective and efficient pedagogical instructions as well as structuring of the teaching curriculum based on the student's profile. The rests of this paper is organized as follows. Design approach of intelligent tutoring system known as CPLITS is discussed in Section 2. Section 3 gives an overview of the proposed architecture of CPLITS with brief description of its components, as well as structuring of the Bayesian Network implanted in the system. The last section provides concluding remarks and future directions for our work.

II. Design approach of an intelligent tutoring system

Intelligent tutoring systems are designed in a way that in their structure incorporate techniques of the Artificial Intelligence - AI in order to provide tutors who will know how to teach, what to teach and how to adequately make knowledge transfer to the student. Examples of ITSs [4], [5], [6], [7], [8] present various ways of dealing with the problem of tutoring using the computer, suggesting different approaches. The main characteristic that increases effectiveness to the task of teaching with an ITS, is system adaptability to the student. Assessing the user state of knowledge and profile requires uncertainty reasoning [4]. Each student has their own characteristic which influences their understanding to interpret learning material. Students receive knowledge in different ways, including hearing and seeing; by reflecting and acting; reasoning either logically or intuitively; by memorizing or visualizing and drawing analogies; and, either steadily or in small bits and large pieces [9]. So, the students must be provided with material models that meet with their personality and their previous knowledge.

Several researchers have tried to show the power of adaptability of the system [10], [11]. For example, a student in an adaptive educational system will be given a presentation that is adapted to his knowledge of the subject, and a suggested set of most relevant links to proceed further [12].

Taking everything mentioned above into consideration, we are proposing a model of intelligent tutoring system – CPLITS. CPLITS is intended for students who want to start learning the key concepts of the C# programming language. The architecture of the proposed model, including its main components and sub-components, as well as modeling the Bayesian Network that will be implemented in

system's architecture. The proposed system is a web-based system.

III. Architecture of CPLITS

Intelligent tutoring systems are characterized by the fact that they store three basic kinds of knowledge: domain knowledge, knowledge about learners and pedagogical knowledge. This knowledge types define the main modules of the system's architecture: the domain knowledge module, the student knowledge module and the pedagogical module. So, the architecture of the proposed system is composed of the following modules: (1) student module; (2) knowledge base module; (3) teaching module; and two additional modules: (4) user interface module that act as a communication bridge between the system and the student; (5) Bayesian Network module; each of which is decomposed to submodules. The architecture of CPLITS is a client-server architecture. The system is web-based and can be accessed from anywhere, at any time using any of the popular web-browsers. The client part contains the user interface for establishing interaction between the student and the system while the server part contains the remaining modules of the system. Figure 1. Shows the full representation of the architecture.

The student module accumulates all of the relevant data related to the student and it contains several submodules: student profile; database with personal information storage; knowledge mastery values/learning style and activity recorder. The student's profile is the main figure that has function to accumulate all the necessary information obtained from the other submodules so that the final result would be creating report for the student with records such as personality information, progression of the student as he\she manages through the learning materials, learning style, overall time spent on the topic, test results and so on. Each generated profile is unique and features a different learning style defined by Felder-Silverman model [9] for categorizing students based on their learning style.

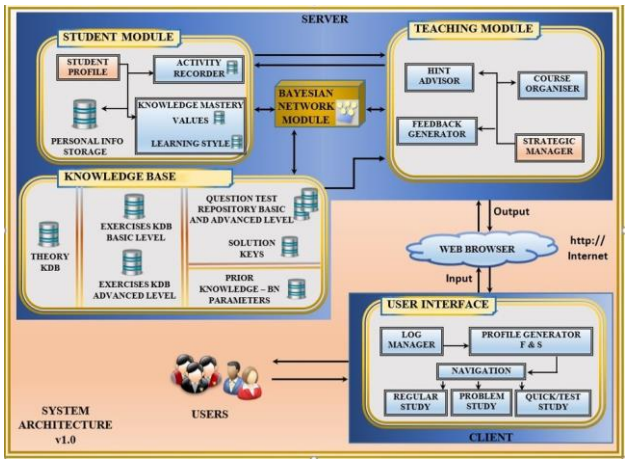

Fig. 1. Representation of the general architecture of CPLITS with their modules and submodules

The knowledge base module covers the key basic concepts needed for understanding the bases of C# programming language regarding the console application. The student will have chance to get introduction in the programming language, data types, operators, variables, control structures, functions, complex data structures and classes. This module is divided into: theoretical materials of various visual formats like text documents, presentations, videos; practical assignments both basic and advanced level; question test repository with different difficulty level of questions; solution keys and prior knowledge that is a base for initializing the parameters in the Bayesian Network.

The teaching module is in charge of leading the student through the learning process and deciding what materials to present to the student regarding the generated learning style. This module contains: strategic manager; feedback generator; hint advisor and organizer of the teaching curriculum. The strategic manager is the main component that coordinate others submodules with function to analyze the structure of the content of each learning concept by establishing communication with the knowledge base, providing helpful instructions and recommendations in solving a particular problem. It also provides up-to-date feedback and as well as generating tests from the test database, organizing the theory and the practical assessments.

The user interface module is the communication bridge between the student and the system. Potential users in the system are: students and administrator. The administrator has full control of the system and is responsible for maintaining the system, creating learning materials, defining the parameters of the Bayesian Network from previous experiences that the students have shown during the study of the matter. On the other hand, when the students get in touch with the system it needs to make signing with user account through the log manager in order to create its own profile. Afterwards learning style is required to be created by answering the questions proposed in the Felder-Silverman test in order to make categorization of students in one of the learning style regarding the organizing of theory materials [13] as shown in Table 1.

Table 1. Different categories of learning styles

Category	Learning style	Preferred format
1	Reflective, visual and sequential	Multimedia lectures
2	Reflective, intuitive and verbal	Theoretical lectures - text
3	Active, sensing and global	Case study with practical tasks

After defining the learning style the student can proceed with learning a particular concept through navigation of the learning materials. Each concept represents a learning package consisted of theory part, practical assignments and test. In other word the student will need to pass three stages of the learning process: (1) regular study that covers the theory, (2) problem study that covers the practical part with different level of difficulty and (3) quick test study that covers evaluation of student's knowledge by answering questions in generated test with different difficulty level regarding the student's knowledge. By the end of each stage the system will present to the student two options before he can continue to next concept: 1- understand the concept, 2-don't understand, repeated. Additionally, it will present some recommendations in order to improve the learning process for the future concepts.

DOI:10.20544/AIIT2018.P15

The Bayesian Network module is consisted of Bayesian model for organizing and navigating through learning materials represented as nodes and arc between nodes.

A. Application of Bayesian Networks in CPLITS

Bayesian Networks - BN are graphical models i.e. directed acyclic graphs for reasoning under uncertainty. As a graphical model of probability, it represents casual relationships of probability between set of random variables and their conditional dependencies.

BN as AI technique in ITS are used for various purposes such as modeling the student's module in order to help students in personalized learning environment, for structuring teaching strategies in order to provide effective and efficient pedagogical guidance for the student in the learning process, structuring of teaching materials for more flexible and easier access, monitoring, evaluating and updating the level of student's knowledge and preferences. BN in CPLITS are used for structuring of the problem from a particular domain, for tracking student's knowledge and guiding the student how to learn and what to learn next. The proposed model of BN for CPLITS it's shown in Figure 2 and it's developed in GeNIe Modeler, software solution for representing BN.

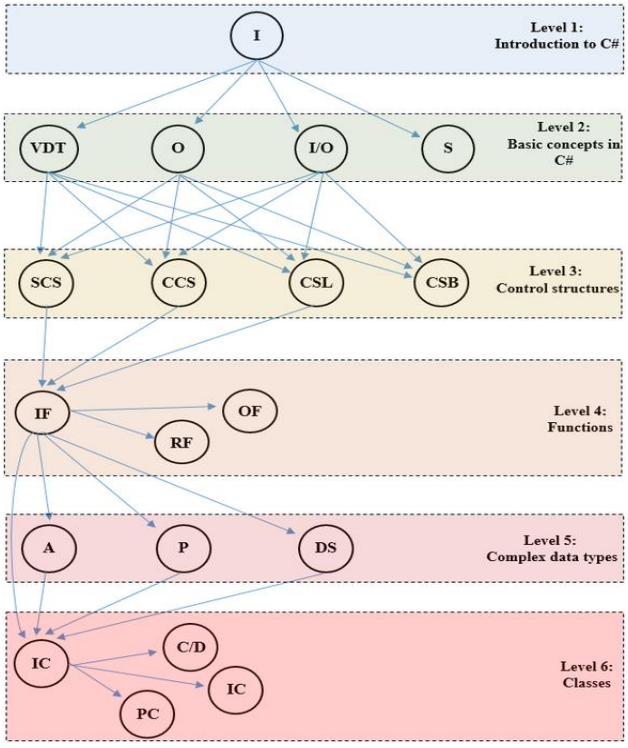

Fig. 2. Full representation of 6-th level BN for navigation

The structure of the Bayesian network consists of nodes and connections that represent the interdependences between. The total number of nodes in the network is 19 nodes with 31 connections between the nodes. Adding the directed arc from one node to another is done in order to determine whether the knowledge of the previous concept is a prerequisite for learning the next concept. The nodes in the BN are organized in six levels depending on the contents of the materials that the student will learn: (1) Introduction in programming, (2) Basic concepts in C#, (3) Control structures, (4) Functions, (5) Complex data structures and (6) Classes. Because of this organization, the BN can be known as 6-th level BN for navigation through learning materials of the C# programming language. Each of these levels corresponds to one of the knowledge mastery values. First and second level correspond to beginner with some knowledge in the area, third and fourth correspond to intermediate level of knowing the area, and the last two levels correspond to advanced level of knowing the area. After defining the structure of the BN next step is to calculate the conditional probabilities between the nodes in the network. For that purpose, it is required to create conditional probability tables - CPT that shows the probability for two or more conditionally dependent nodes – events. The formula of Bayes' Theorem for updating of the values in the network is going to be used for calculating the probabilities.

Considering that the nodes in the network are defined and the conditional probability tables are fulfilled from the parameters from the prior knowledge database, the network can be updated in situation where no observations are made of any event, i.e. the nodes takes the initial values of probabilities they have. Each node has their own probability and the occurrence state of every node can be "mastered" and "not mastered" that refers to the concept. To make it easier for the student to navigate through the materials, CPLITS marks the learning concepts with appropriate colors and signs in terms of which concepts are learned and which are not learned. Such material markings are generated from the BN in the system and can be categorized according to the three criteria: (1) mastered concepts, (2) concepts that can be learned and (3) concepts that are not available for learning. The markings for the concepts can be of the following character: green color with tinted sign, blue color with unlocked padlock and grey color with locked padlock appropriately for mastered concepts, concepts that can be learned and concepts that are not available for learning. This marking is done in order to help the student to gain an insight for the current knowledge level of each concepts and guidance for what he\she can learn further. Furthermore, a concept can be categorized as mastered only if the BN show that the probability value P (concept = mastered | evidence) is greater or equal to 0.7, where evidence is a variable for the student's knowledge of the previous learning concept. A concept can be categorized as concept that can be learned only in situation where all previous concepts of the previous level are marked as mastered concepts with probability greater or equal to 0.7. Finally, a concept can be categorized as concept not available for learning only in situation when there is at least one prerequisite concept that is not mastered.

After everything is defined we can calculate the probability that the student has mastered the concept - variables and data types before any evidence is observed i.e. prior probability. For that purpose, CPTs are presented in Table 2 and Table 3.

Table 2. Given CPT for introduction in programming and variables and data types

Introduction in programming - I	
Mastered	0.71
not mastered	0.29

Table 3. Given CPT for introduction in programming and variables and data types

	Introduction in programming - I	mastered	not mastered
Variables and data types - VDT	mastered	0.72	0.21
	not mastered	0.28	0.79

Based on the above data from the CPTs we can calculate the probability for variables and data types such as:

$$P[VDT = mastered] = P[VDT = mastered \mid I = masterd] * P\,(I = matered)$$

$$+ P(VDT = mastered \mid I = not\ mastered) * P\,(I = not\ mastered)$$

$$= 0.72 * 0.71 + 0.21 * 0.29 = 0.57 * 100 = 57\% \quad (1)$$

The obtained results show that 57% or 0.57 is the probability that the student is prepared to master the concept variables and data types. The calculated results for the rest nodes in the network are shown in Figure 3.

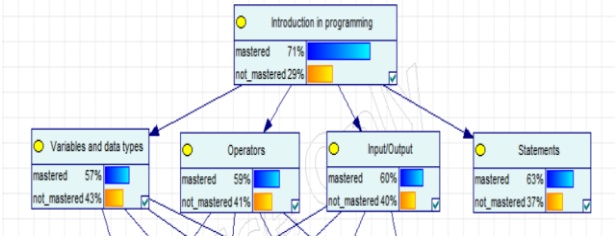

Fig. 3. Updated values of probabilities in BN without observing any evidence

In the same way the posterior probability can be calculated i.e. probability after certain nodes-evidences are observed. For the abovementioned example, we can calculate the probability that the student has mastered variables and data types after mastering the concept introduction to programming. The formula has the following form:

$$P[VDT|I] = \frac{P[VDT, I]}{P[I]}$$

$$= \frac{P[VDT = mastered|I = mastered]P[I = masterd]}{P(I = masted)}$$

$$= \frac{0.72*0.71}{0.17} = \frac{0.5112}{0.71} = 0.72 * 100 = 72\% \quad (2)$$

The results for the computed posterior probabilities are shown in Figure 4.

It can be concluded that 72% is the probability that the student masters the concept variables and data types by previously mastering introduction to programming.

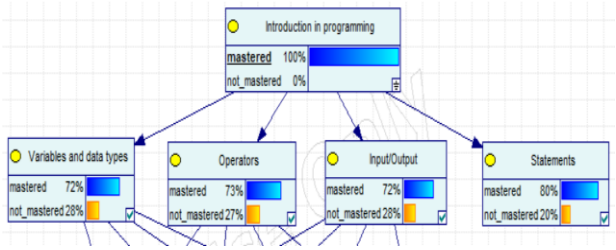

Fig. 4. Updated values of probabilities in BN with observing evidence

In the same way, the probabilities can be computed for the rest of the nodes. By using GeNIe Modeler these computations can be done automatically.

IV. Conclusion

Nowadays the effectivity of the education is mostly due to the development of information and communication technologies in order to improve the knowledge transfer to students. ITS as one link from the chain of information technologies have found adequate application in education. In this paper it has been developed a model of ITS that will aid student to better understand the concepts of C# programming language by giving them support in the navigation through learning materials how to learn and what to learn in the upcoming concepts. The beneficiaries of using this system will allow to student to learn at a distance with time and costs savings. It also provides them much more personalization and adaptability, user-friendly interface that follow the concept one-size fits all and the most important thing increase the level of effective and efficient motivation in the learning process. In the future, the efforts would be concentrated on implementing the system in a final software solution by writing code, updating the model in terms of expanding the system's functionalities, adding new learning concepts and implementing the Bayesian Network via the SMILE platform offered by BAYES FUSION, an independent library based on C #, Java, .NET and Python.

References

[1]. J. R. Anderson, and B. J. Reiser, "The LISP tutor", *Byte, 10*(4), 1985, pp.159-175

[2]. B. P. Woolf, "Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning", Burlington, MA, USA: Morgan Kaufmann, 2008

[3]. J.D. Fletcher, "Research Foundations for the Advanced Distributed Learning Initiative", Institute for defense analyses- Advanced Distributed Learning Center for Intelligent Tutoring Systems Research & Development, 2010

[4]. H. Gamboa, and A. Fred, "Designing intelligent tutoring systems: a Bayesian approach", ICEIS 2001 - Artificial Intelligence and Decision Support Systems, 2001, pp.452-458

[5]. A. E. Permanasari, I. Hidayah, and S. Nugraha, "A Student Modeling Based on Bayesian Network Framework for Characterizing Student Learning Style", Journal of Computational and Theoretical Nanoscience · October 2014, DOI: 10.1166/asl.2014.5702, Advanced Science Letters Vol. 20, 2014, pp.1936-1940

[6]. K. G. Schulze, R. N. Shelby, D. J. Treacy, M. C. Wintersgill, K. Vanlehn, and A. Gertner, "An intelligent tutor for classical physics", 2000, *The Journal of Electronic Publishing, University of Michigan Press*. http://www.press.umich.edu/ jep/06-01/schulze.html

[7]. C. J. Butz, Sh. Hua, and R. B. Maguire, "A web-based Bayesian intelligent tutoring system for computer programming", Web Intelligence and Agent Systems, 4(1), 2006, pp.77–97

[8]. Y. Wang, and J. Beck, "Class vs. Student in a Bayesian Network Student Model", K. Yacef et al. (Eds.): AIED 2013, LNAI 7926, pp. 151–160, 2013. Springer-Verlag Berlin Heidelberg 2013

[9]. R. Felder, and L. Silverman, "Learning and Teaching Styles", Journal of Engineering Education, 78 (7), 1988, pp.674–681

[10]. Y. Wang, and N.T. Heffernan, "The Student Skill Model", In: Cerri, S.A., Clancey, W.J., Papa- dourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, 2012, pp. 399–404. Springer, Heidelberg

[11]. Z.A. Pardos, and N.T. Heffernan, "Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing" In: De Bra, P., Kobsa, A., Chin, D. (eds.) UMAP 2010. LNCS, vol. 6075, 2010, pp. 255–266. Springer, Heidelberg

[12]. P. Brusilovsky, "Adaptive hypermedia. User modeling and user-adapted interaction", 11(1-2), 2001, pp.87–110

[13]. Blaženska-Tabakovska, N., Ivanović, M., Klašnja-Milićević, A., and Ivković, J. " Comparison of E-Learning Personalization Systems: Protus and PLeMSys", International Journal: Emerging Technologies in Learning, iJET, vol.12, no.1, 2017, pp.57-70.

International Conference on Applied Internet and Information Technologies, 2018
DOI:10.20544/AIIT2018.P16

Trends in software maintenance support tasks: A study in a micro software company

Zeljko Stojanov* and Jelena Stojanov*
* University of Novi Sad / Technical faculty "Mihajlo Pupin", Zrenjanin, Serbia
zeljko.stojanov@uns.ac.rs, jelena.stojanov@uns.ac.rs

Abstract - Support tasks in software maintenance relates to tasks that help in ensuring proper functioning of software applications without modifying them. These tasks include training, consultation, clients' data management, installation and configuration of software applications. The aim of this study is to investigate trends in support maintenance tasks performed in a micro software company. Tasks' related data are extracted from the company internal repository of tasks. Trend analysis is based on descriptive statistical methods and proposed support task typology that reflects the real maintenance practice in the company. The analysis relates to trends in working hours spent by programmers on solving support tasks and distribution of tasks among senior and junior programmers. The results of trend analysis are useful for planning and decision-making activities in the selected company. Benefits for the company, limitations of the study, and research implications are outlined.

I. INTRODUCTION

Software maintenance is the costliest part of software life cycle, consuming over 50% of all costs [1,2,3,4]. Even more, maintenance costs for software systems being used for long time greatly exceed software development costs [5]. Although software maintenance has been recognized as very important for productivity and efficiency of software organizations [6], maintenance tasks are treated as short term tasks that should be completed as fast as possible [3]. In addition, maintenance tasks are considered among software engineers as less interesting and boring comparing to software development tasks, which often causes avoidance of these tasks [7].

Software maintenance has been researched over 40 years. In this period, several classifications and typologies of maintenance types have been proposed in software engineering literature. The first and the most influential typology in software maintenance is defined in 1976 by Swanson [8]. It includes corrective, adaptive and perfective tasks. This typology is extended with preventive maintenance in Standard for Software Engineering-Software Maintenance, ISO/IEC 14764 [9]. Chapin et al. [10] proposed more detailed classification of maintenance tasks based on the modifications implemented on software applications (software, code, customer-experienced function). This extended typology includes 12 types of tasks grouped in 4 clusters: support interface, documentation, software properties, and business rules. Support interface includes tasks such as trainings, consultations and evaluations of software systems. Due to the high importance of software modifications in software maintenance (software properties and business rules clusters), support interface is categorized as less important, and therefore much less researched segment of software maintenance practice. Regardless of the maintenance task type, software maintenance tasks should have minimal impact on clients' business performance [11].

In the book *Software Engineering Best Practices* [12] Jonas identified customer support as one of the main cost drivers in software life cycle since it is highly labor intensive and generally unsatisfactory. Kitchenham et al. [13] proposed a maintenance ontology aimed at identifying factors that influence software maintenance, according to which software organizations should organize and manage support activities into well-defined roles performed by skilled staff. This ontology proposes organization of maintenance support in the following three levels: (1) Non-technical help desk staff engaged for logging problems and providing the first assistance to software users, (2) Technical support that communicate with end users, understand their problems and can suggest some quick solutions, and (3) Maintenance (software) engineers that do modifications on software products.

Software maintenance management requires careful and on time management activities to reduce errors and ensure software usefulness for end users[14]. One of the most reliable ways to detect problems and potential improvements in software maintenance is to collect data in a real industrial setting and conduct data analysis to identify trends in everyday maintenance practice. Efficient and reliable trend analysis in software maintenance is based on [15]: (1) well defined software maintenance processes, (2) tracking of software maintenance requests on daily basis, (3) precisely defined procedures for collecting relevant data, and (4) validated data that are extracted for the proposed measurement goals.

Based on the above discussion, investigation of support tasks trends in software maintenance is proposed as the aim of this study. The rest of the paper is organized as follows. Related work section outlines studies related to trends in software maintenance and justifies the need for researching support maintenance tasks in industrial practice. The third section presents the study aimed at investigating trends in maintenance support tasks in a local micro software company in Serbia. The fourth section presents discussion of benefits for the company, limitations of the study, and research implications for industry practitioners and researchers. The last section contains concluding remarks and brief reference to future research directions.

II. RELATED WORK

Trend analysis in software maintenance is used for detecting issues in everyday industrial practice. It requires collecting reliable data in an appropriate period, which are suitable for achieving proposed research goals and are aligned with the goals of organization involved in the research [16]. However, the use of traditional management techniques is not completely applicable in software maintenance due to specific internal organization of software companies and non-material nature of software products. This section outlines some empirical studies dealing with trend analysis in software maintenance and justifies the need for inquiring trends in software maintenance support tasks.

Kenmei et al. [17] organized an empirical study aimed at investigating trends of change requests based on time series. Data were extracted from version control and bug tracking systems for open source software projects Eclipse, JBoss and Mozilla. Created model enables prediction of number of change requests to be received in next 2 to 6 weeks, which is important for project staffing and planning.

April [15] presented an empirical study with trend analysis of supply and demand of software maintenance services. Maintenance demand is defined as a total of all received maintenance requests, while supply is total of all services given to customers. The study was conducted in Integratik, an ERP development firm in Canada, as a part of software process improvement project. The aim of a developed trend model is to help in managing customer expectations from software maintenance. Trend analysis is performed in a way that enable identification of trends related to personnel workload and trends for delivered software systems.

Bando and Tanaka [18] presented trend analysis of accidents occurred in financial information systems. Data collection included several sources: newspapers, news release on websites, magazines and books. Trend analysis of accidents was performed according to type, severity and faults. Trend analysis identified the following types of accidents: service related, processing related, information related, and cybercrime related. The results of trend analysis revealed that human made faults are increasing more than physical faults. Based on results, the authors proposed several priority issues for dependability improvements of cases such as repeated accidents, frequent accidents, human-made mistakes, or software quality.

Trend analysis aimed at identifying trends in maintenance request processing in a very small software company is presented in [19]. Trend analysis is used within software process improvement project, with the main objective to identify segments of maintenance practice that should be improved. Data analysis was used for identifying monthly trends for number of maintenance requests, number of spent working hours on solving requests, distribution of requests per maintenance types (tasks). For data analysis is used maintenance task typology developed in the company. As one of the results of maintenance processes improvement project, a new task typology was developed by considering the context in the company and relevant scientific literature [20]. Trend analysis aimed at investigating distribution of maintenance tasks among programmers according to a newly proposed typology is presented in [20].

Aggarwal et al. [21] conducted a study aimed at investigating trends for Chromium Browser Project. The trends relate to bugs extracted from Issue Tracking Systems (ITS) for the selected software. Each bug report contains the following fields: Priority(Pri), Category(Cr), Operating System(OS), Milestone(M) and Type. Trends are used for discovering topics in bug descriptions that mostly attract developers, which enables discovering of duplicated bugs and reopened bugs. The results of trend analysis are useful for expertise modeling, resource allocation and knowledge management.

This short literature review revealed that trend analysis is used for investigating software maintenance practice. The common objectives are to identify trends in the selected segment of practice and to propose improvements or to estimate future trends. Presented studies deal with business software applications purposely developed for specific group of clients [15,18,19,20] or with open source software projects [17,21], considering general maintenance trends or trends related to corrective tasks. However, there are no identified studies dedicated to support tasks in software maintenance. The study presented in this paper aims at filling this gap in research literature and contributing to empirical knowledge base in the field of software maintenance.

III. CASE STUDY

The study was carried out in an indigenous software company focused on producing business software applications for local clients in Serbia. According to *User guide to the SME Definition* published by European Commission [22], the company can be classified as a micro enterprise since it has 7 employees (6 programmers and 1 technical secretary). In total, 48 software applications are used by over 100 clients in Serbia. The study objective is to identify trends in supportive maintenance performed in the company and based on the results to propose directions for improving everyday practice.

A. Software maintenance trends

Data extracted from the company internal repository of tasks were used for data analysis. The data were extracted from the repository by using SQL scripts, and after that they are imported in MS Excel for further processing and analysis. Analyzing trends requires data collected from a longer period, which ensures identification of trends relevant for the practice to be investigated [23]. Data analysis is based on 2293 software maintenance tasks performed in the period of 19 months, starting from February 2013. According to detailed investigation of maintenance tasks' trends [20], 2036 tasks were classified as maintenance tasks, which is 88.79% of all tasks.

Classification of maintenance tasks was performed based on a new typology introduced in [20]. The following types of maintenance tasks were included in the typology: adaptation, correction, enhancement, preventive, and support. Based on the data analysis presented in [20], 467 support tasks were identified, which is 22.94% of all maintenance tasks. Since support tasks relates to providing